Elmar Schömer
Ann-Christin Wörl

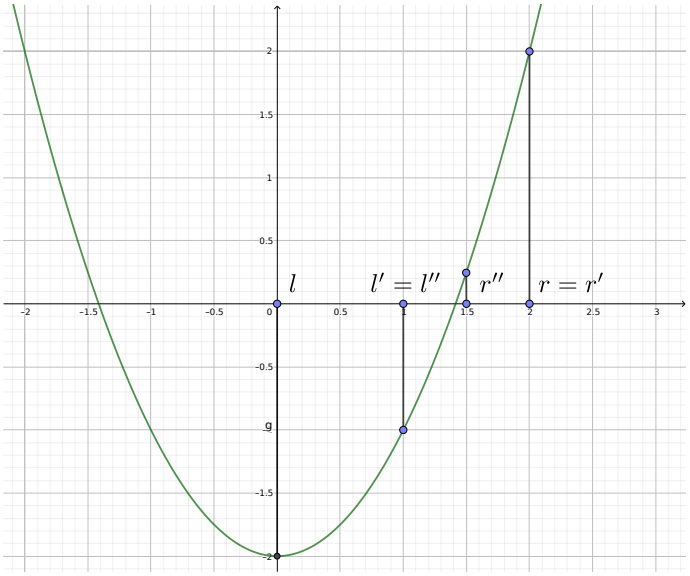
3. Übungsblatt

Abgabe: Dienstag, der 14.11.2023, 14:00 Uhr

Aufgabe 1: Bisektion (20 Punkt)

Wir wollen mit Hilfe des Intervallhalbierungsverfahrens (siehe https://de.wikipedia.org/wiki/Bisektion) eine Nullstelle z einer Funktion f approximativ bestimmen. Wir starten mit einem Intervall $[l, r]$, von dem wir aufgrund des Vorzeichenwechsels von $f(l)$ und $f(r)$ wissen, dass sich darin die Nullstelle z befinden muss. Wir bestimmen den Mittelpunkt m des Intervalls $[l, r]$, berechnen das Vorzeichen von $f(m)$ und ersetzen diejenige Intervallgrenze mit dem selben Vorzeichen durch die neue Grenze m. In diesem Teilintervall setzen wir nun unsere Suche nach der Nullstelle z fort. Diesen Prozess wiederholen wir solange, bis die Länge unseres aktuellen Suchintervalls kleiner als ein vorher definierter Wert ϵ ist. Die folgende Grafik visualisiert diesen Prozess noch einmal anhand eines Funktionsgraphen und der mathematischen Beschreibung:

Gesucht z, so dass $f(z) = z^2 - 2 = 0$



Sei $[l, r] = [0, 2]$
$f(l) < 0$ und $f(r) > 0$
$\Rightarrow z \in [l, r]$
$m = \frac{l+r}{2} = 1$
Da $f(m) < 0$, setze
$[l', r'] = [m, r] = [1, 2]$

$\downarrow$

Sei $[l', r'] = [1, 2]$
$f(l') < 0$ und $f(r') > 0$
$\Rightarrow z \in [l', r']$
$m' = \frac{l'+r'}{2} = 1.5$
Da $f(m') > 0$, setze
$[l'', r''] = [l', m'] = [1, 1.5]$

$\downarrow$

$\vdots$

Schreiben Sie ein Python-Programm zur Approximation von $\sqrt{2}$ als Nullstelle der Funktion $f(x) = x^2 - 2$.

```
1  l = 0.0
2  r = 2.0
3  eps = 1e-6
```

Aufgabe 2: Pascalsches Dreieck (5+10+5 Punkt)

Wir wollen die Einträge des Pascalschen Dreiecks (siehe https://de.wikipedia.org/wiki/Pascalsches_Dreieck) zeilenweise berechnen (ohne Verwendung von Feldern und Rekursion). Die Einträge des Pascalschen Dreiecks sind durch die Binomialkoeffizienten gegeben:

$$\binom{n}{k} = \frac{n \cdot (n-1) \cdots (n-k+1)}{1 \cdot 2 \cdots k}$$

```
1
1  1
1  2   1
1  3   3   1
1  4   6   4    1
1  5  10  10    5    1
1  6  15  20   15    6   1
1  7  21  35   35   21   7   1
1  8  28  56   70   56  28   8  1
1  9  36  84  126  126  84  36  9  1
```

1. Begründen Sie, warum das folgende Python-Programm diese Binomialkoeffizienten nur unter Verwendung von ganzzahliger Arithmetik korrekt berechnet.

```
1  n = 5
2  k = 3
3  i = 0
4  b = 1
5  while i < k:
6      b = b * (n-i)
7      b = b // (i+1)
8      i = i + 1
9  print(b)
```

2. Ergänzen Sie das obige Programm so, dass es die n-te Zeile des Pascalschen Dreiecks ausgeben kann.

3. Geben Sie nun die ersten 10 Zeilen des Dreiecks aus.

Aufgabe 3: Pythagoreisches Tripel (5+5+10 Punkt)

In der Zahlentheorie besteht ein Pythagoreisches Tripel aus drei verschiedenen natürlichen Zahlen, bei denen die Summe der Quadrate der beiden kleineren Zahlen gleich dem Quadrat der größten Zahl ist (siehe https://de.wikipedia.org/wiki/Pythagoreisches_Tripel). Das folgende Programm sucht ein solches pythagoreische Tripel im Bereich kleiner 20.

```
1  c = 5
2  while c < 20:
3      a = 1
4      while a < c:
5          b = a+1
6          while b < c:
7              if a*a+b*b == c*c:
8                  print(a,b,c)
9              b += 1
10         a += 1
11     c += 1
```

1. Verändern Sie das obige Programm so, dass Sie die gefundenen pythagoreischen Tripel nicht mehr ausgeben, sondern nur noch zählen. Wie viele solche Tripel (a, b, c) mit $a < b < c < 1000$ gibt es?

2. Warum hat Ihr Programm eine so hohe Laufzeit?

3. Verändern Sie Ihr Programm so, dass es sehr viel schneller die gesuchte Zahl der Tripel bestimmen kann.

Aufgabe 4: Risiko II (20 Punkt)

Wir betrachten noch einmal das Würfelexperiment beim Risiko Spiel (siehe Übung 2, Aufgabe 4), bei dem der Angreifer mit drei Würfeln gegen zwei Würfel des Verteidigers antritt. Da fünf faire Würfel zum Einsatz kommen, gibt es $6^5 = 7776$ gleich wahrscheinliche Würfelereignisse. Laut Wikipedia führen 2890 davon zu dem Ergebnis $2:0$, 2611 zu dem Ergebnis $1:1$ und 2275 zu dem Ergebnis $0:2$.
Schreiben Sie ein Python-Programm, das diese Statistik bestätigt. Verzichten Sie möglichst auf die Verwendung zusätzlicher Bibliotheken.

Aufgabe 5: Fibonacci (5+5+5+5 Punkt)

Die Fibonacci-Folge (siehe https://de.wikipedia.org/wiki/Fibonacci-Folge) beschreibt eine unendliche folge natürlicher Zahlen, bei der jede Zahl der Summe ihrer beiden vorangehenden Zahlen entspricht, beginnend mit den ersten Fibonacci-Zahlen $f_0 = 0$ und $f_1 = 1$. Sie ist wie folgt rekursiv definiert:

$$f_n = \begin{cases} 0 & \text{für } n = 0 \\ 1 & \text{für } n = 1 \\ f_{n-1} + f_{n-2} & \text{für } n \geq 2 \end{cases}$$

n	0	1	2	3	4	5	6	7	...
f_n	0	1	1	2	3	5	8	13	...

1. Schreiben Sie ein Python-Programm, das die ersten 100 Fibonacci-Zahlen $f_0, f_1, \ldots, f_{99}$ ausgibt.
2. Verifizieren Sie mit Hilfe Ihres Programms, dass für alle berechneten Folgeglieder Folgendes gilt:

$$f_n = \frac{1}{\sqrt{5}} \left(\frac{1+\sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left(\frac{1-\sqrt{5}}{2} \right)^n$$

   Rundungsfehler der Fließkommaarithmetik dürfen Sie ignorieren.
3. Für alle $n \in \mathbb{N}$ gilt:

$$\sum_{i=0}^{n-1} f_i = f_{n+1} - 1$$

   Verifizieren Sie diese Aussage für $n = 99$.
4. Bestimmen Sie für $n < 100$ alle Fibonacci-Zahlen, die Primzahlen sind (siehe https://de.wikipedia.org/wiki/Fibonacci-Primzahl).